Package ‘Rbeast’

July 26, 2019

Type Package

Version 0.2.1

Date 2019-07-22

Title Bayesian Change-Point Detection and Time Series Decomposition

Author Kaiguang Zhao [aut, cre],
Tongxi Hu [aut],
Yang Li [aut]

Maintainer Kaiguang Zhao <lidar.rs@gmail.com>

Depends R (>= 2.10.0),methods, utils

Description Interpretation of time series data is affected by model choices. Different models can give different or even contradicting estimates of patterns, trends, and mechanisms for the same data--a limitation alleviated by the Bayesian estimator of abrupt change,seasonality, and trend (BEAST) of this package. BEAST seeks to improve time series decomposition by forgoing the ``single-best-model'' concept and embracing all competing models into the inference via a Bayesian model averaging scheme. It is a flexible tool to uncover abrupt changes (i.e., change-points), cyclic variations (e.g., seasonality), and nonlinear trends in time-series observations. BEAST not just tells when changes occur but also quantifies how likely the detected changes are true. It detects not just piecewise linear trends but also arbitrary nonlinear trends. BEAST is applicable to real-valued time series data of all kinds, be it for remote sensing, economics, climate sciences, ecology, and hydrology. Example applications include its use to identify regime shifts in ecological data, map forest disturbance and land degradation from satellite imagery, detect market trends in economic data, pinpoint anomaly and extreme events in climate data, and unravel system dynamics in biological data. Details on BEAST are reported in Zhao et al. (2019).

LazyData true

NeedsCompilation yes

RoxygenNote 6.0.1

Repository CRAN

Date/Publication 2019-07-26 08:10:02 UTC

R topics documented:

`avhrr_YellowStone` *30 years' AVHRR NDVI data at a Yellostone site*

Description

`avhrr_YellowStone` is a vector comprising 30 years' AVHRR NDVI data at a Yellostone site

Usage

```
data(avhrr_YellowStone)
```

Source

Rbeast v0.2.1

Examples

```
library(Rbeast)
data(avhrr_YellowStone)
plot(avhrr_YellowStone,type='l')

result=beast(avhrr_YellowStone)
plot(result)
```

`beast` *Bayesian estimation of abrupt changepoint, nonlinear trend, and periodicity*

Description

Apply a Bayesian model averaging algorithm called "BEAST" to decompose time series data into three contrasting components: Abrupt changes, trends, and cyclic/seasonal variations.

Usage

```
beast(data, option=list(),demoGUI=FALSE,...)
```

Arguments

- `data` — a vector or matrix of input data. Missing values are allowed and can be indicated by `NA`, `NaN`, or a customized value specified in the 2nd argument `option` (e.g., `option$omittedValue=-9999`). If `data` is a vector of dimension `Nx1` or `1xN`, it is treated as a single time series of length `N`. If `data` is a matrix of size `NxM`, it is considered as multiple time series: The row dim "`N`" is the time-series length; the col dim "`M`" is the number of time series. For earth sciences or remote sensing applications, `data` can be a 3D array made of stacked time-series images.
- `option` — (optional). If present, `option` can be either an INTEGER specifying the known period of the cyclic/seasonal component or a LIST specifying various paramaters for the BEAST algorithm. The "`period`" parameter must be an INTEGER specifying the number of samples/values per cycle (e.g, a monthly-sampled time series with an annual period has a period of 12, that is, `option$period=12`). Other possible parameters are demonstrated below in Example 3 of the Examples Secction. If `option` is absent, BEAST will use default model parameters; in particular, the `period` of the cyclic component of time series will be best guessed via auto-correlation before running BEAST.
- `demoGUI` — a boolean indicator. If set to `TRUE`, BEAST will be run in a GUI demostration mode, with a GUI window to show an animation of the MCMC sampling in the model space step by step. Note that "`demoGUI=TRUE`" works only for Windows x64 systems not Windows 32 or Linux/Mac systems.
- `...` — additional parameters, not used currently but reserved for future extension

Value

The output is an object of class "beast". It is a list, consisting of the following components. In the explanations below, the input `data` is supposed to be of size `NxM`: `N` is the time series length and `M` is the number of time series (`M>=1`):

- `time` — a vector of size `1xN`: the times at the `N` sampled locatons. By default, it is simply set to `1:N`
- `sN` — a vector of size `1xM`. `sN` gives the mean number of seasonal changepoints for each of the M time series. If `data` is a single time series (i.e., `M=1`), `sN` will be a scalar.
- `tN` — a vector of size `1xM`. `tN` gives the mean number of trend changepoints for each of the M time series. If `data` is a single time series, `tN` will be a scalar.
- `sNProb` — a matrix of size `(option$maxKnotNum_Season+1)xM`. For the i-th time series (i.e.,`1=<i<=M`),`sNProb[,i]` gives a probability distribution of having a certain number of changepoints over the range of [0,option$maxKnotNum_Season]; for example, `sNProb[1,i]` is the probability of having no seasonal changepoint in the ith time series; `sNProb[3,i]` is the probability of having 2 changepoints (i.e, 3-1=2).
- `tNProb` — a matrix of size `(option$maxKnotNum_Trend+1)xM`. For the i-th time series (i.e.,`1=<i<=M`), tNProb[,i] gives a probability distribution of having a certain number of trend changepoints over the range of [0,option$maxKnotNum_Trend];

for example, `tNprob[1,i]` is the probability of having no trend changepoint in the ith time series; `tNprob[4,i]` is the probability of having 3 changepoints (i.e, 4-1=3).

`sProb`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `sProb[,i]` gives a probability distribution of having a seasonal changepoint at a certain time over the time range of [1,N]. Plotting `sProb[,i]` will depict a curve of probability-of-being-changepoint over the time for the i-th time series.

`tProb`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `tProb[,i]` gives a probability distribution of having a trend changepoint at a certain time over the time range of [1,N]. Plotting `tProb[,i]` will depict a curve of probability-of-being-trend-changepoint over the time for the i-th time series.

`s`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `s[,i]` gives the best fitted seasonal component.

`t`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `t[,i]` gives the best fitted trend component.

`b`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `b[,i]` gives the estimated slope in the fitted trend component over time.

`sSD`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `sSD[,i]` gives the standard deviations of the fitted SEASONAL component.

`tSD`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `tSD[,i]` gives the standard deviations of the fitted TREND component.

`bSD`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `bSD[,i]` gives the standard deviations of estimated SLOPE.

`sCI`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `sCI[,i]` gives the 95% credible intervals of the fitted SEASONAL component.

`tCI`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `tCI[,i]` gives the 95% credible intervals of the fitted TREND component.

`bCI`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `bCI[,i]` gives the the 95% credible intervals of the estimated SLOPE.

`horder`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `horder[,i]` gives the estimated harmonic order used to approximate the SEASONAL component over time.

`torder`
: a matrix of size `NxM`. For the i-th time series (i.e., `1=<i<=M`), `torder[,i]` gives the estimated polynomial order used to approximate the TREND component over time.

`scp`
: a matrix of size `(option$maxKnotNum_Season)xM`. For the i-th time series (i.e., `1=<i<=M`), scp[,i] gives the most possible locations of changepoints in the the SEASONAL component.

`tcp`
: a matrix of size `(option$maxKnotNum_Trend)xM`. For the i-th time series (i.e., `1=<i<=M`), tcp[,i] gives gives the most possible locations of changepoints in the TREND component.

References

Zhao K, Wulder M, Hu, T, Bright R, Wu Q, Qin H, Li Y, Toman E, Mallick B, Zhang X, and Brown M (2019). Detecting change-point, trend, and seasonality in satellite time series data to track abrupt changes and nonlinear dynamics: A Bayesian ensemble algorithm. Remote Sensing of Environment (accepted manuscript)

Examples

```
library(Rbeast)

# A MODIS time series of NDVI for a forest plot in Ohio. It has 23 samples
# per year (i.e., period=23). Note that the input time series to "beast" must
# be spaced/observed at regular time intervals, with missing values indicated
# by NAs, NaNs, or a custimized value (see Example 3). Iregular-sampled time
# series data need to be first aggegrated at a regular time interval of your
# choice before running beast; if not, beast may give meaningless results or,
# even worse, terminate abnormally.
data(modis_ohio)
plot(modis_ohio)


#---------------------------------Example 1--------------------------------#
# No "option" argument supplied below so default parameters are used. The period
# (i.e., 23) will be estimated via auto-correlation. Letting the program compute the
# period of a cyclic time series is not always reliable, so it is always suggested
# to directly supply the period as in Example 2 and Example 3.
out=beast(modis_ohio)
plot(out)
#*****************************End of Example 1****************************#


#---------------------------------Example 2--------------------------------#
# "option" is set to 23, specicfying the period of modis_ohio as 23
out=beast(modis_ohio,23)
plot(out)
plot(out$s)       #The same as plot(out$s[,1]): plot the seasonal curve
plot(out$sProb)   #Plot the probability of observing seasonal changepoints
plot(out$t)       #The same as plot(out$t[,1]): plot the trend
plot(out$sProb)   #Plot the probability of observing trend changepoints

#*****************************End of Example 2****************************#

#---------------------------------Example 3--------------------------------#
# Specify the option parameters explicilty
opt=list()              #Create an empty list to append individual model parameters

opt$period=23           #Period of the cyclic/seasonal component of the modis time series
opt$minSeasonOrder=2    #Min harmonic order allowed in fitting season component
opt$maxSeasonOrder=8    #Max harmonic order allowed in fititing season component
opt$minTrendOrder=0     #Min polynomial order allowed to fit trend (0 for constant)
opt$maxTrendOrder=1     #Max polynomial order allowed to fit trend (1 for linear term)
```

```
opt$minSepDist_Season=20#Min seperation time btw neighboring season change-pts(mustbe >=0)
opt$minSepDist_Trend=20 #Min seperation time btw neighboring trend change-pts(must be >=0)
opt$maxKnotNum_Season=4 #Max number of season changepoints allowed
opt$maxKnotNum_Trend=10 #Max number of trend changepoints allowed
opt$omittedValue=-999   #A customized value to indicate bad/missing values in the time
                        #series, in additon to those NA or NaN values.

opt$printToScreen=1     #If set to 1, display some progress status while running
opt$printCharLen=150    #The length of chars in each status line when printToScreen=1

# The following parameters used to configure the reverisible-jump MCMC (RJMCC) sampler
opt$chainNumber=2       #Number of parallel MCMC chains
opt$sample=1000         #Number of samples to be collected per chain
opt$thinningFactor=3    #A factor to thin chains (e.g., samples taken every 3 iterations)
opt$burnin=500          #Number of burn-in samples discarded at the start of each chain
opt$maxMoveStepSize=30  #For the move proposal, the max window allowed in jumping from
                        #the current changepoint
opt$resamplingSeasonOrderProb=0.2 #The probability of selecting a re-sampling proposal
                                  #(e.g., resample seasonal harmonic order)
opt$resamplingTrendOrderProb=0.2  #The probability of selecting a re-sampling proposal
                                  #(e.g., resample trend polynomial order)

opt$seed=65654          #A seed for the random generator: If seed=0,random numbers differ
                        #for different BEAST runs. Setting seed to a chosen non-zero integer
                        #will allow reproducing the same result for different BEAST runs.
opt$computeCredible=0   #If set to 1, compute 95% credible intervals: The results will be
                        #saved as sCI, tCI, and bCI in the output variable.
opt$fastCIComputation=0 #If set to 1, employ a fast algorithm to compute credible intervals
opt$computeSlopeSign=1  #If set to 1, compute the probability of having a postive slope in
                        #the estimated trend. The result will be saved as bsign in the outpu
                        #variable.

opt$computeHarmonicOrder=1 #If set to 1, compute the mean harmonic order of the fitted
                           #seasonal component. The result will be saved as "horder" in
                           #the output variable.
opt$computeTrendOrder=1    #If set to 1, compute the mean polynomial order of the fitted
                           #trend component. The result will be saved as "torder" in
                           #the output variable.


#opt$outputToDisk=0        #(if set to 1, results will be written to files in a folder)
#opt$outputFolder ='c:/out'#Specify the output folder when outputToDisk=1
#opt$lengthPerTimeSeries_infile=300#the time series length if input data come from a binary


# Use "opt" defined above in the beast function. Note that to run beast(), not all the indiv
# parameters in option need to be explicitly specified. If an parameter is not given in opti
# default value will be used.
out=beast(modis_ohio, opt)
plot(out)

#******************************End of Example 3****************************#
```

```
#--------------------------------Example 4--------------------------------#
# Run an interactive GUI to visualize how BEAST is samplinig from
# the possible model spaces in terms of the numbers and timings of
# seasonal and trend changepoints. The GUI inferface allows changing
# the option parameters interactively. This GUI is only available on
# Win x64 machines, not Mac or Linux.
beast(modis_ohio, 23, demoGUI=TRUE)
#*****************************End of Example 4****************************#

#--------------------------------Example 5--------------------------------#
# 'simdata' is a 300x3 matrix, consisting of three time series
data(simdata)

# Plot individual time series. As a toy example, all the three time series
# are the same.
plot(simdata[,1])
plot(simdata[,2])

# Below, the option is defined in the command line as a temporary list.
out=beast( simdata, list(period=24, chainNumber=3, sample=1000, burnin=200) )

# "out" contains results for the three time series. Plot the result for the second one
plot(out,2)
#*****************************End of Example 5****************************#
```

`modis_ohio` *14 years' MODIS EVI data at a pixel in Southern Ohio*

Description

`modis_ohio` is a vector comprising 14 years' MODIS EVI data at a pixel in Southern Ohio.

Usage

```
data(modis_ohio)
```

Source

Rbeast v0.2.1

Examples

```
library(Rbeast)
data(modis_ohio)
```

```
plot(modis_ohio,type='l')
result=beast(modis_ohio)
plot(result)
```

---

plot.beast *Changepoint Detection*

Description

Plot the result obtained from the beast function.

Usage

```
  ## S3 method for class 'beast'
plot(x, index, ...)
```

Arguments

`x`	x must be an object of class "beast". It is the returned result from the beast function.
`index`	If x contains results for multiple time series. `indx` specifies which of them is plotted.
`...`	further arguments passed to the `plot` function.

Value

This function creates various plots to demonstrate the results of a beast decomposition. .

Examples

```
library(Rbeast)
data(simdata)
result=beast(simdata)
plot(result,1)
plot(result,2)
```

simdata *Simulated time series to test BEAST*

Description

`simdata` is a 300 x 3 matrix, consiting three time series of length 300. Currently, the three time series are the same. It is used to illustrate BEAST can handle multiple time series at a singel function call. of BEAST.

Usage

```
data(simdata)
```

Source

Rbeast v0.2.1

Examples

```
library(Rbeast)
data(simdata)
plot(simdata,type='l')

result=beast(simdata)
plot(result,1)
plot(result,2)
plot(result,3)
```